САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Кафедра математической теории игр и статистических решений

Югов Илья Игоревич

Выпускная квалификационная работа бакалавра

Исследование особенностей свойств контуров на разных видах растровых изображений

Направление 010400.62
Прикладная математика и информатика

Научный руководитель,
кандидат физ.-мат. наук,
доцент
Ковшов А.М.

Санкт-Петербург
2016

Содержание

Введение ... 3
Цель ... 5
Глава 1. Контурный анализ ... 6
1.1 Бинаризация изображения ... 6
1.2 Получение контуров ... 7
1.3 Получение свойств контуров ... 8
1.3.1 Пример работы программы ... 13
Глава 2. Проверка достоверности подхода ... 15
2.1 Исследование первое ... 15
2.2 Исследование второе ... 16
2.3 Вывод ... 18
2.4 Программная реализация ... 18
Глава 3. Классификация изображений ... 22
3.1 Кластерный анализ ... 23
3.1.1 Кластеризация методом k-means ... 24
3.2 Кластеризация контуров ... 25
3.2.1 Программная реализация ... 26
3.3 Нейронные сети ... 32
3.3.1 Персептрон ... 33
3.3.2 Классификация изображений нейронной сетью ... 35
3.4 Существующие решения ... 38
Заключение ... 40
Список литературы ... 41

Введение

Каждый день в мире делается огромное количество новых фотоснимков. Люди фотографируют ландшафтные пейзажи, городские достопримечательности, лица друзей, природные памятники и многое другое. Значительная часть этих снимков попадает в интернет, однако алгоритмы поиска по графическим изображениям до сих пор не достигли желаемого уровня. Для ускорения поиска можно приписывать каждому изображению набор отличительных признаков, например, присутствие на картинке характерных объектов: домов, водоёмов, деревьев, людей. Также можно различать изображения по наличию крупных или мелких деталей, присутствию перспективы, контрастных границ однородных цветовых полей, характерных текстур и других свойств. В представленной работе исследуется возможность отнести растровое изображение к заданному типу при помощи анализа свойств контуров, которые можно выделить на растровом изображении, приведя его к бинарному виду.

Решение текущей задачи важно, для компьютерного зрения, о чём говорится во многих работах, например, в [1], [2], [4], данная наука появилась относительно недавно и очень быстро развивается, растёт спрос на специалистов в данной области. Множество методов уже применяется на практике, например – камеры на дорогах фиксирующие превышение скорости, сканеры штрих кодов, кодирование адресов сайтов в QR – коды и многие другие [1], [2].

Проводимое исследование будет полезно в задачах распознавания и компьютерного зрения, результаты исследования так же можно использовать как предобработку перед сегментацией какого-либо объекта, а также для поиска тематических изображений в интернете.

Задача данного исследования: научится распознавать и различать друг от друга изображения водной поверхности, лесных массивов, городской местности, фотографии горных массивов и полей.

Подобные задачи, как правило решают сверточными [6-10] нейронными сетями, такой способ требует больших вычислений, для качественных изображений обучение такой сети занимает достаточное количество времени [6, 8-10], т.к. сеть принимает на вход изображение попиксельно, а если изображений порядка 2 тысяч, то время работы сети может длится очень долго. Также для нейронной сети требуется, чтобы вход в сеть был одинаковый для всех примеров [6, 9-10], это означает, что если изображения в выборке разных размеров, то их необходимо сжимать, растягивать, и обрезать, что тоже занимает много времени. Поэтому было решено рассмотреть данную задачу с помощью анализа свойств контуров и использовать классические нейронные сети типа персептрон в качестве классификатора. Таким подходом так же решали задачу распознавания объектов [8-10], в качестве входных данных подавались, как и в свёрточную сеть изображения попиксельно, что не сильно ускорило работу, и ухудшало точность распознавания. В данной работе предлагается посмотреть на другой способ классификации изображений, когда изображение задаётся не всеми его пикселями, а лишь всеми контурами изображения с их свойствами.

Цель

Целью данной работы является нахождение контуров и выявление таких их свойств, с помощью которых будет возможно отнести изображения к соответствующему типу, решая задачу классификации.

Для достижения данной цели было решено:

1. Выявить подходящие свойства контуров
2. Составить задачу классификации
3. Создать программную реализацию алгоритмов
4. Сравнить полученное решение с ранее известными

В данной работе рассматриваются только изображения в высоком качестве, т.к. на данный момент качество съёмки достигло высокого уровня, даже с мобильного устройства получаются изображения в хорошем качестве. Рассматриваются типы однородных изображений (т.е. такие, на большей части которых изображена одна тематика).

Глава 1. Контурный анализ

Контурный анализ используется при обработке изображений и распознавания образов, с его помощью можно хранить, описывать и осуществлять поиск объектов на изображении, представленных в виде внешних и внутренних очертаний – контуров. Изображения, получаемые различной техникой могут содержать множество помех, шумов, искажений чёткости и контраста, что искажает информацию на нём. Такое часто проявляется в полностью автоматизированных системах, а требуется производить от 10^8 до 10^{14} и более вычислений в секунду. Один из подходов решающим данные трудности, является обработка изображения не попиксельно, а только обрабатывая его контуры. Такой подход слабо зависит от яркости и цвета фотоснимка, устойчив к смене датчика, делающего снимок, не зависит от времени съемки [1,2].

1.1 Бинаризация изображения

Важную роль в контурном анализе играют бинарные изображения, т.к. они намного меньше занимают места в памяти, чем цветные, а потеря информации незначительна [1].

Переход от цветного изображения к бинарному осуществляется множеством способов [1]:

a) Методом Оцу
b) Методом Янни
c) Методом среднего
d) По средней яркости

Для своей работы был использован метод по средней яркости изображения. Пусть, $I_1(x, y)$ – функция яркости исходного изображения, $I_2(x, y)$ – функция яркости бинарного, I_m – средняя яркость фотоснимка, тогда $I_2(x, y)$ вычисляется по формуле (1.1)

$$I_2(\mathrm{x},\mathrm{y}) = \begin{cases} 1\,, & \text{если } I_1(\mathrm{x},\mathrm{y}) \geq I_m \\ 0\,, & \text{если } I_1(\mathrm{x},\mathrm{y}) \leq I_m \end{cases} \quad (1.1)$$

1.2 Получение контуров

Алгоритмы поиска контуров описаны в множестве работ, их подробное описание можно найти в работах [1] [2]. Для понимания, что такое контур, какими они бывают и как находятся, рассмотрим пример изображения на рисунке 1.1. На данном рисунке представлен вариант изображения с одной фигурой отмеченной черными пикселями, белые пиксели – фон изображения. Алгоритм поиск контура на изображении: 1) Идём построчно по белым пикселям с левого верхнего угла изображения, до первого появления чёрного пикселя (пиксель с зелёной точкой на рисунке 1.2). 2) Обход границы фигуры ограниченной чёрными пикселями против часовой стрелки, до того момента, пока мы не ввернёмся к пикселю, с которого начинался обход (пиксель с зелёной точкой на рисунке 1.2). 3) Аналогично обходим внутренний контур, обход происходит по часовой стрелке. На рисунке 1.2 два контура одной фигуры, зелёный и жёлтый, внешний и внутренний соответственно, таких вложенностей может быть много, для простоты здесь она одна.

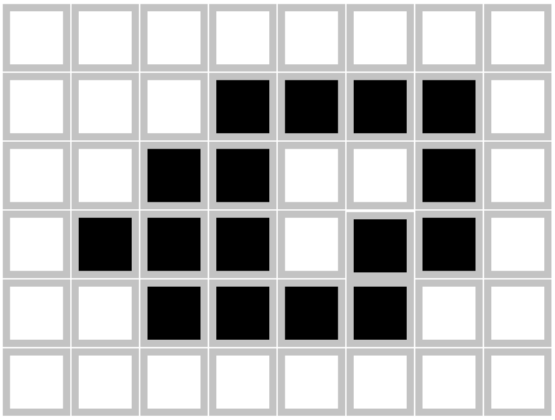

Рисунок 1.1 Пример простого контура

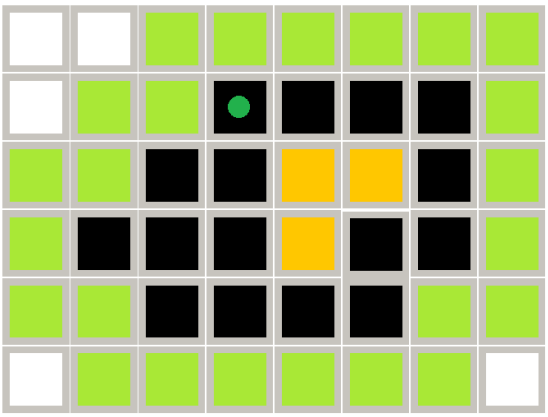

Рисунок 1.2 Зелёная точка – начальная точка обхода внешнего контура. Зелёная ломанная – внешний контур, желтая – внутренний.

Таким образом получается, что контур на изображении, это набор пикселей, соответствующий замкнутой границе фигуры охватываемой

контуром. Если рассмотреть эту фигуру, то у неё можно найти такие параметры, как: площадь, периметр, центр масс, и др. Такие параметры будем называть свойствами контура, о которых будет сказано подробнее в следующей главе.

1.3 Получение свойств контуров

Поиск большинства свойств контуров осуществляется во время нахождения самих контуров (это сделано для уменьшения времени работы программы), за исключением тех, которые зависят от других свойств. Например, свойство отношения главных моментов инерции, если моменты инерции вычисляются по ходу поиска контура, то их отношение рассчитывается после того, когда контур найден.

Вычисляемые свойства контуров:

1. Минимальное значение горизонтальной координаты.
2. Максимальное значение горизонтальной координаты.
3. Минимальное значение вертикальной координаты.
4. Максимальное значение вертикальной координаты.
5. Горизонтальный диаметр – разность между максимальным и минимальным значениями горизонтальной координаты.
6. Вертикальный диаметр – разность между максимальным и минимальным значениями вертикальной координаты.
7. Горизонтальная координата центра тяжести.
8. Вертикальная координата центра тяжести.
9. Максимальный диаметр – наибольшее расстояние между двумя точками контура (Из-за долгого вычисления для больших контуров эта величина заменена удвоенным максимальным радиусом).
10. Максимальный радиус – наибольшее расстояние между центром тяжести и точкой на контуре.

11. Длина контура.
12. Площадь фигуры, охватываемой контуром.
13. Угол направления главной оси инерции фигуры, охватываемой контуром. Здесь главной осью считается та, которая соответствует наибольшей вытянутости контура, то есть перпендикулярная оси, дающей наибольший момент инерции (далее β).
14. Главный набольший момент инерции фигуры, охватываемой контуром (далее I_X).
15. Главный наименьший момент инерции фигуры, охватываемой контуром (далее I_Y).
16. Полярный момент инерции фигуры, охватываемой контуром сумма двух главных моментов инерции.
17. Отношение двух главных моментов инерции (меньшего к большему) (далее R_I).
18. Отношение максимального радиуса к длине контура (далее R_D).
19. Отношение площади фигуры, охватываемой контуром к квадрату длины контура (далее R_L).
20. Отношение квадрата площади к полярному моменту инерции фигуры, охватываемой контуром (далее R_S).

Абсолютно все свойства нам не важны, важны лишь те, которые являются инвариантными при равномерном растяжении, сжатии и повороте контура, это свойства: R_I, R_D, R_L, R_S. Так же было решено использовать одно не инвариантное свойство β, т.к. в процессе исследований было выявлено, что водная поверхность и городская местность хорошо различаются по этому параметру (Об этом будет сказано более подробно в пункте 2.2).

Более подробно о некоторых свойствах.

1) I_X, I_Y – Главные наибольший и наименьший момент инерции.

Это геометрические моменты инерции по площади фигуры, охватываемой контуром. Дают соответственно наибольшее и наименьшее

значения момента инерции при вращении фигуры вокруг осей X и Y соответственно, где оси совпадают с нижней и левой границами изображения. В общем виде геометрический момент считается по формуле (1.2):

$$J_{S_a} = \int_{(S)} r^2 dS \quad (1.2)$$

интегрирование выполняется по поверхности S,

dS — элемент поверхности,

r — расстояние от dS до оси a [3].

Т.к. в данной задаче моменты инерции считается численно, тогда, если x_i и y_i – координата i-ого пикселя контура и i = 1…n, то их можно вычислить по формуле (1.3):

$$I_X = \sum_{i=1}^{n} {x_i}^2,\ I_Y = \sum_{i=1}^{n} {y_i}^2 \quad (1.3)$$

2) R_I – Отношение двух главных моментов инерции.

Как видно из (Рисунка 1.3), данный параметр отвечает за вытянутость контура вдоль осей инерции (чем контур более вытянутый, там отношение меньше) [3].

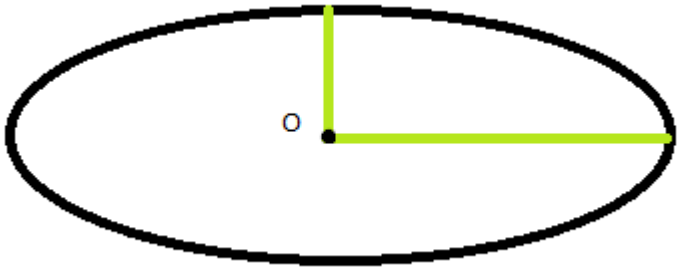

Рисунок 1.3 Чёрный эллипс – контур. Зелёные прямые - оси инерции. O – центр масс.

3) R_D – Отношение максимального радиуса к длине контура.

R_L – Отношение площади фигуры, охватываемой контуром к квадрату длины контура.

Данные параметры отвечают за “лохматость” контура и вычисляются по формулам (1.4):

$$R_D = \frac{R}{L},\ R_L = \frac{S}{L^2} \quad (1.4)$$

R – максимальное расстояние от центра масс, до точки на границе контура

L – длина контура.

S – Площадь контура.

Различие между двумя свойствами поясним на рисунках контуров с одинаковой площадью 1.4, 1.5, 1.6. “лохматостью” называем свойство контура, влияющее на изменение длины контура при неизменной площади. С помощью данного параметра можно будет отличить контуры изображённые на рисунках 1.4, 1.5, 1.6. Все эти контура имеют одинаковую площадь, но различную длину, если на первом рисунке длина будет наименьшей, при фиксированном расстоянии от центра масс контура, от на остальных рисунках она будет увеличиваться, и самая наибольшая длинна будет на рисунке 1.5. Таким образом получается, что с помощью данных свойств можно различить эти 3 варианта контура.

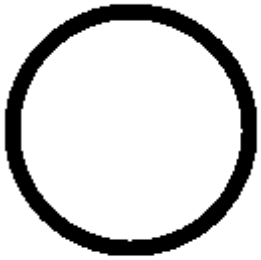

Рисунок 1.4 Контур в виде окружности

Рисунок 1.5 Более лохматый контур чем на Рисунке 1.4

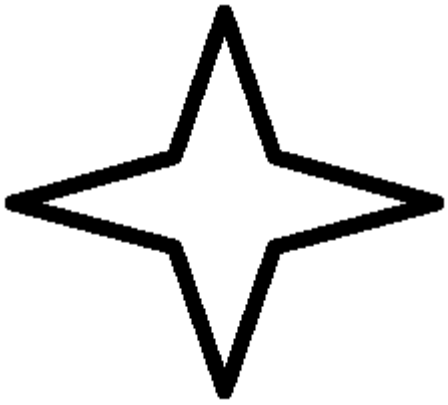

Рисунок 1.6 Лохматый контур

4) R_S – Отношение квадрата площади к полярному моменту инерции фигуры, охватываемой контуром.

Полярный момент – в общем виде вычисляется по формуле в полярной системе координат:

$$J_{S_a} = \int_A \rho^2 dA$$

Т.к. в нашей задаче начало декартовой С.К совпадает с полюсом полярной С.К., то момент может быть вычислен по формуле:

$$J_{S_a} = I_X + I_Y \quad (1.5)$$

Тогда получается, что (1.5) преобразуется к виду (1.6)

$$R_S = \frac{S^2}{J_{S_a}} = \frac{S^2}{I_X + I_Y} \quad (1.6)$$

Данный параметр отвечает за разброс основной массы контура, от его центра масс, это можно заметить, если рассмотреть фигуры с одинаковой площадью и длиной, но различными моментами инерции, как показано на рисунке 1.7.

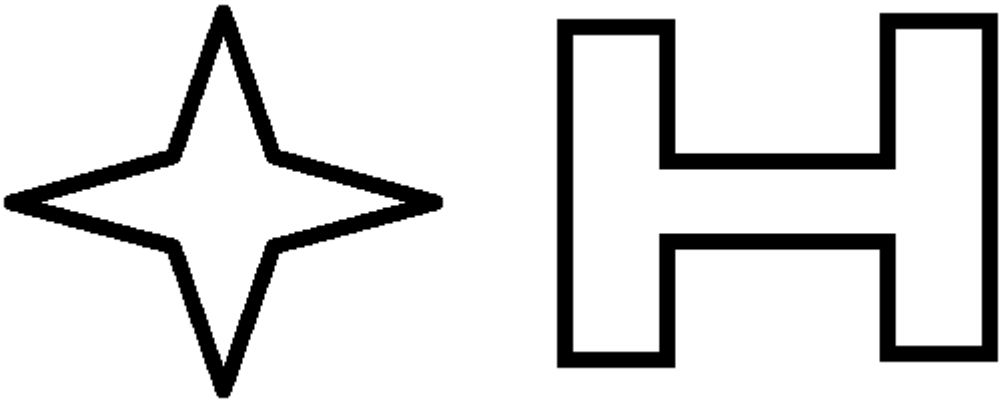

Рисунок 1.7 Контура с одинаковой площадью и длинной

1.3.1 Пример работы программы

Для знакомства с контурным анализом и некоторыми методами обработки изображений моим научным руководителем мне была предоставлена программа, которая походу работы была дополнена некоторыми модулями. В данном приложения осуществлялась работа с изображениями, находились контура, их свойства, тестировались различные фильтры по свойствам контуров, выводились некоторые закономерности о которых будет сказано в главе 2.

Рассмотрим пример работы программы, при нахождении контуров и их свойств на примере изображения на рисунках 1.8 и 1.9. Перейдя на главном экране в Texture Contour Control, выбрав метод бинаризации by average (по средней яркости) и нажав Make Contours программа вычислит все контура, где зелёный цвет означает, что контур внешний (т.е при его нахождении обход осуществлялся против часовой стрелки о чём было сказано в предыдущей главе), оранжевый – внутренний. Если нажать на контур, снизу будут выведены его основные свойства и отобразятся оси инерции на контуре (Рисунок 1.10).

В данном приложении есть возможность работы непосредственно со свойствами контуров, для этого необходимо перейти в окно Contour Statistics (Рисунок 11). Снизу на данном экране можно выбрать свойства по которым необходимо сделать фильтрацию, которую затем визуализировать в окне Texture Contour Control (Об это подробнее в параграфе 2.4).

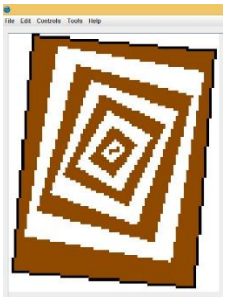

Рисунок 1.8 Исходное изображение

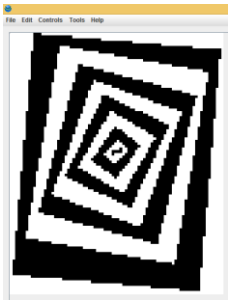

Рисунок 1.9 Бинариованное изображение 1.8

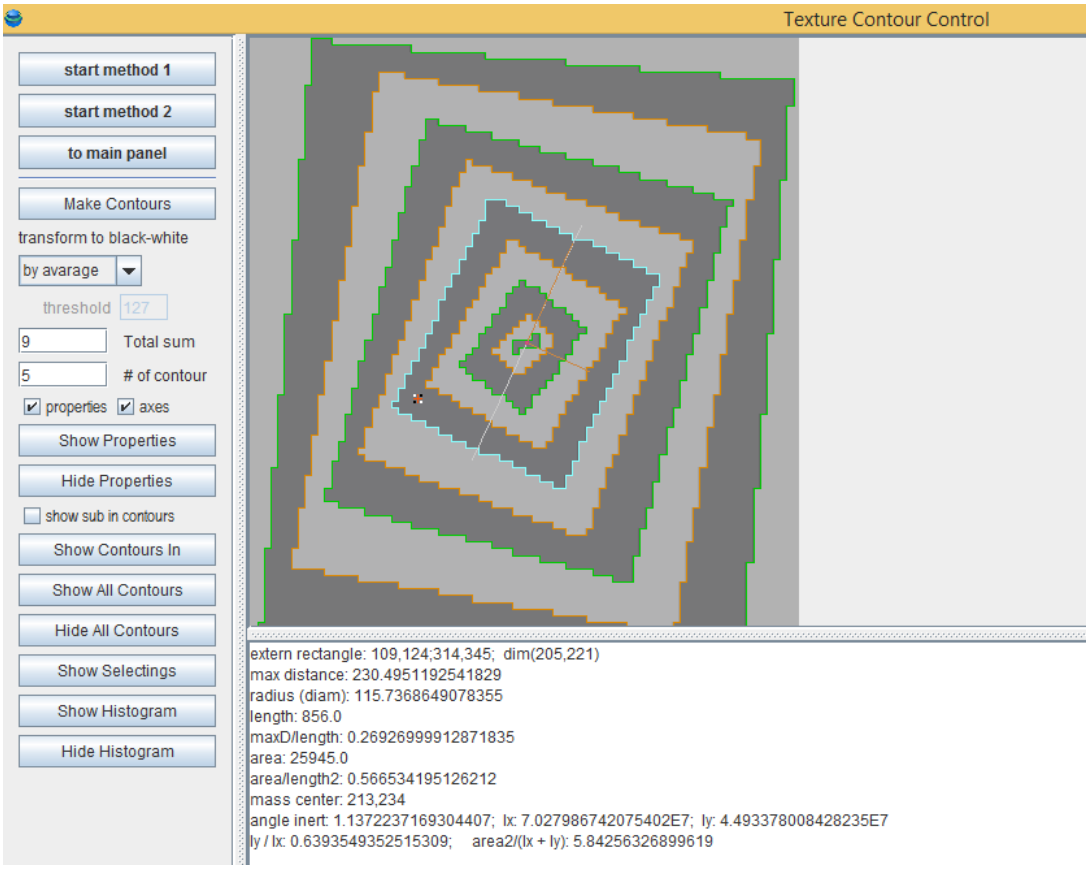

Рисунок 1.10 Пример работы программы. Зелёные контура – внешние, оранжевые – внутренние.

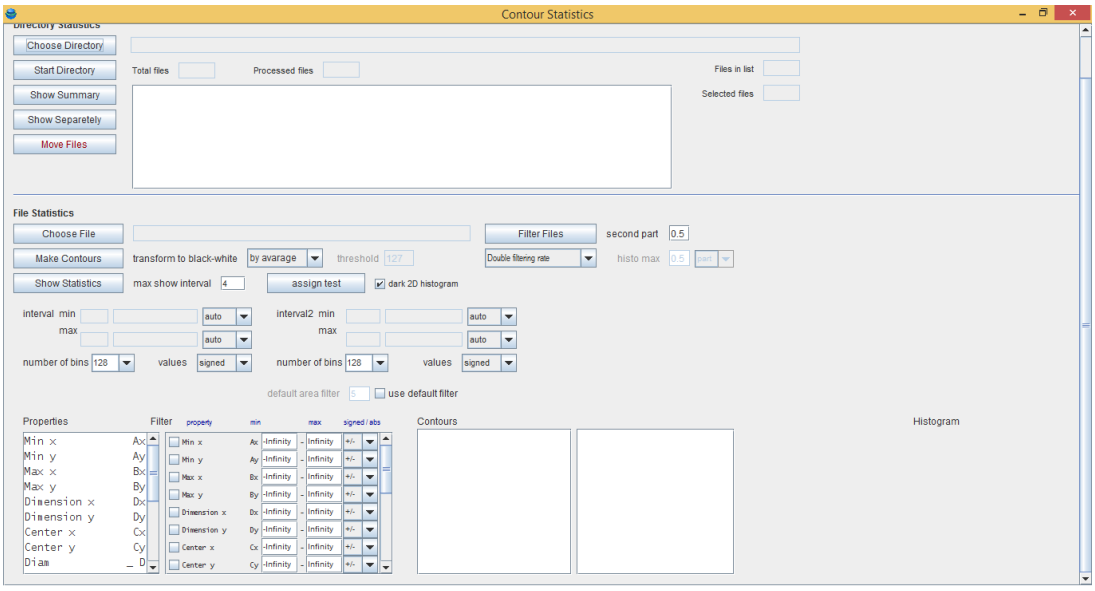

Рисунок 1.11 Окно, для работы со свойствами контуров

Глава 2. Проверка достоверности подхода

В данной главе рассматривается вопрос, возможности ли решить данную задачу с помощью анализа свойств контуров. Было проведено два исследования, направленных на выяснение этого вопроса.

2.1 Исследование первое

Задача:

Выявить свойство или свойства контуров, с помощью которых возможно будет определить, что на фотоснимке определена вода.

В первом исследовании были рассмотрены изображения водной поверхности, и эмпирическим путем было выявлено, что если отобрать контура, площадь которых более 10 пикселей (т.к. данные контура не несут никакой информации, они слишком малы и считаются шумом). После этого, взглянув на гистограмму распределения контуров по β, то для большей части всех контуров изображения воды, β∈ [-0.1; 0.1].

Рисунок 2.1 Водная поверхность

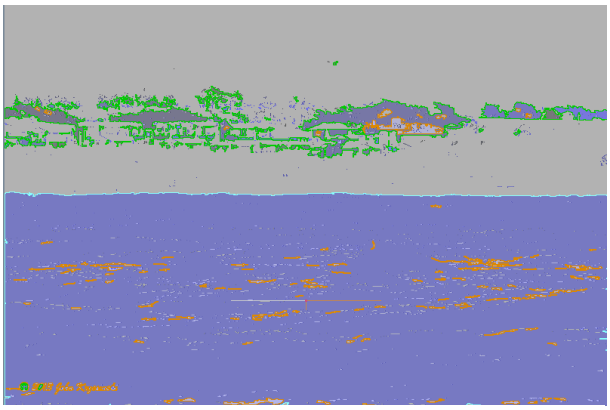

Рисунок 2.2 Бинаризованное изображение.

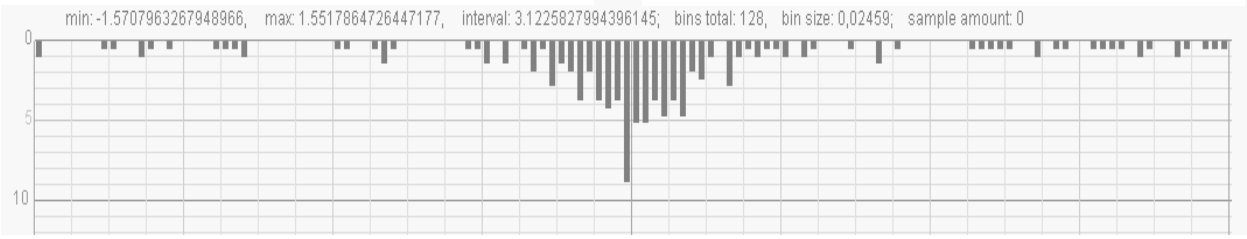

Рисунок 2.3 Гистограмма распределения контуров, по углу инерции. Одно большое деление – 0.4 градуса, маленькое – 0.1 градуса.

Рассмотрим пример работы алгоритма, на рисунках 2.1 и на 2.2 представлено изображение воды, в цветном и бинарном виде, причём на втором выделены контура, для которых S > 10. На рисунке 2.3 видно сосредоточение контуров около нуля по β.

2.2 Исследование второе

Задача:

Выявить параметры контуров, с помощью которых можно будет различать изображения водной поверхности и городском местности.

Были рассмотрены фотоснимки воды, города, и для чистоты эксперимента был взят третий тип, который не должен быть распознан, как первый или второй. В ходе работы, было замечено, что если отобрать контуры, для которых $R_I < 0.1$ и S > 10, то выделяются не пересекающиеся группы изображений воды и города по абсолютному значению β .

Пример работы алгоритма на изображении водной поверхности:

Рисунок 2.5 Водная поверхность

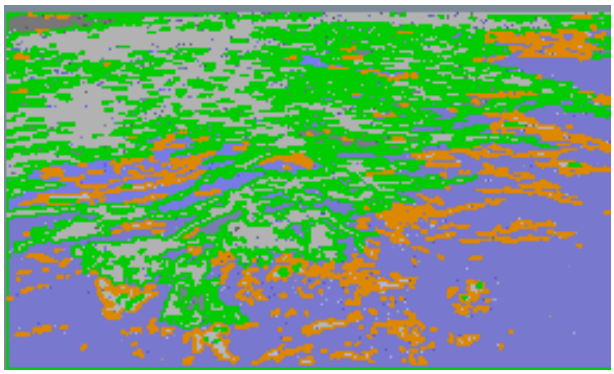

Рисунок 2.4 Бинаризованное изображение водной поверхности с выделенными контурами, прошедшими фильтрацию (Цвет выделения не имеет значения)

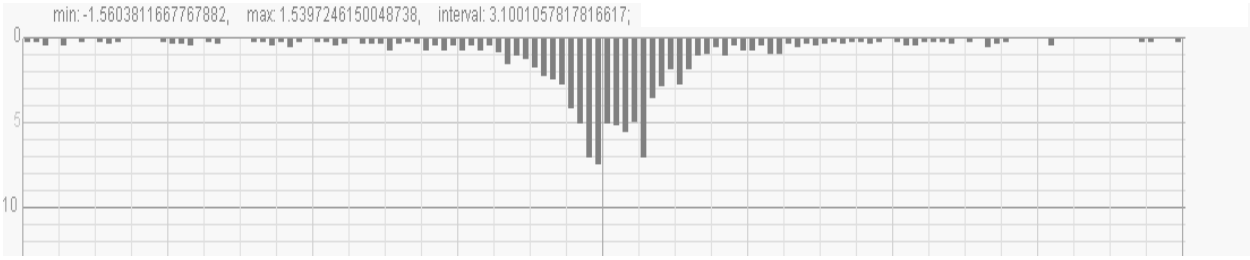

Рисунок 2.6 Гистограмма распределения контуров, по углу инерции. Одно большое деление – 0.4 градуса, маленькое – 0.1 градуса.

Пример работы алгоритма на изображении городской местности:

Рисунок 2.8 Городская местность

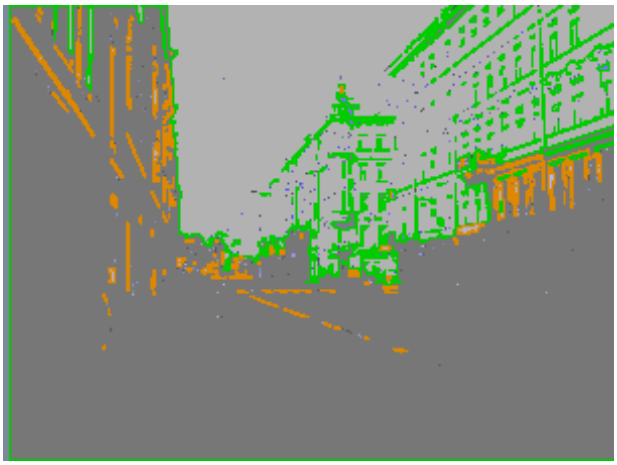

Рисунок 2.7 Бинаризованное изображение городской местности с выделенными контурами, прошедшими фильтрацию (Цвет выделения не имеет значения)

Рисунок 2.9 Гистограмма распределения контуров, по углу инерции. Одно большое деление – 0.4 градуса, маленькое – 0.1 градуса.

На рисунках 2.4, 2.5, 2.6 и 2.7, 2.8, 2.9 можно заметить, что наблюдаются закономерности на гистограммах распределения контуров (рисунок 2.6 для воды и Рисунок 2.9 для города). Зелёный и оранжевый цвет выделения контуров на бинарном изображении создан для того, чтобы можно было на фотоснимке увидеть вложенность контуров (оранжевые – внутренние, зелёные – внешние). Работу алгоритма представлена в таблице 2.1.

Наблюдаемые группы для различных тематик:

- Вода, если $\beta \in (0.03\ ;\ 0.2) \cup (-0.2\ ;\ -0.03)$
- Город, если $\beta \in (0\ ;\ 0.03) \cup (\frac{\pi}{2} - 0.3\ ;\ \frac{\pi}{2})$
- Другое, при других значениях угла

Таблица 2.1 Результат второго исследования

Основные тематики изображений / **Наблюдения**	**Водная поверхность**	**Городская местность**	**Другое**
Вода	65	2	2
Город	6	78	63
Другое	14	8	113
Общее количество фотоснимков	**85**	**88**	**178**
Точность	**76%**	**89%**	**63%**

2.3 Вывод

Из проделанных исследований можно сделать вывод, что данный подход даёт ожидаемые результаты, как в первом, так и во втором экспериментах. Следующим этапом будет автоматизация данного алгоритма, с помощью машинного обучения.

2.4 Программная реализация

Исследования, описанные в данной главе были проделаны в программе, о которой шла речь в параграфе 1.3. Покажем пример того, как находились закономерности из второго эксперимента, на примере изображения водной поверхности (рисунок 2.10). На рисунке 2.11 представлено тот же фотоснимок, но уже в виде контуров. Далее покажем все контура, площадь которых больше 10 и отношение моментов меньше 0.1, для этого необходимо задать фильтр в Contour Statistics (рисунок 2.12). И так выбраны свойства по которым необходимо было сделать фильтрацию, далее посмотрим на эти контура в Texture Contour Control, нажав кнопку Show Selecting (Рисунок 2.13). Далее посмотрим диаграмму распределения контуров по свойству выбранному ранее (Рисунок 2.12 свойство выделено квадратом (номер 2)), на данный момент выбрано свойство угол инерции. Гистограмма распределения показана на

рисунке 2.14, на котором выделена область определённая во втором исследовании для контуров водной поверхности. Таким образом проводились исследования данной главы и выводились закономерности с помощью которых удалось различить водную поверхность от городской местности.

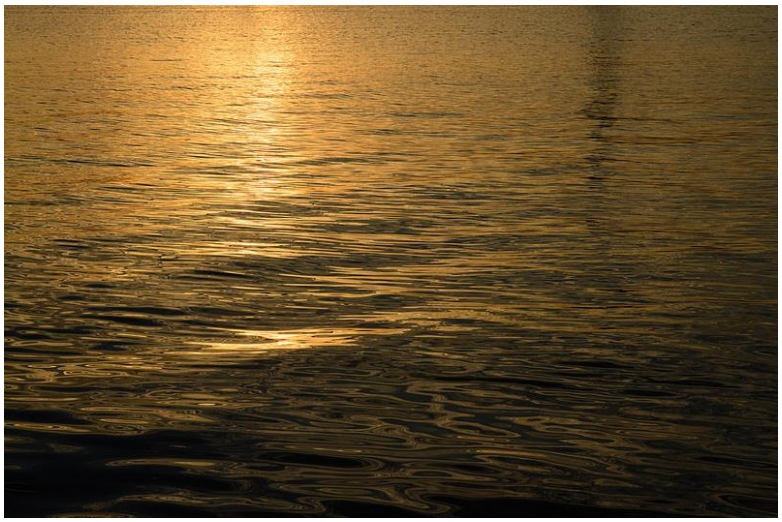

Рисунок 2.10 Изображение водной поверхности

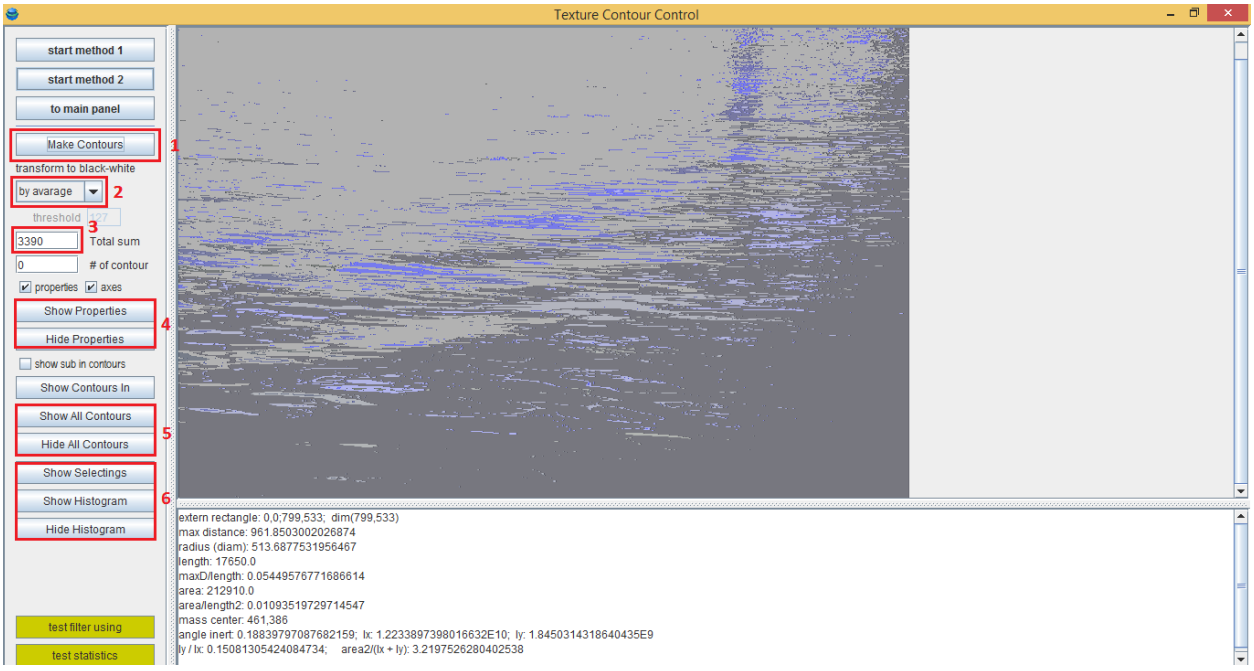

Рисунок 2.11 1 – кнопка, нажав на которую получим изображение контуров в виде контуров, 2 – всплывающий список, в котором можно выбрать тип бинаризации, 3 – текстовое поле в котором показано общее число полученных контуров, 4 – кнопки, с помощью которых можно показать или убрать свойства контуров, 5 – кнопки нажав на которые можно выделить все контура, либо убрать их выделение, 6 – кнопки для работы с фильтрацией контуров по свойствам контуров(фильтр выбирается в Contour Statistics)

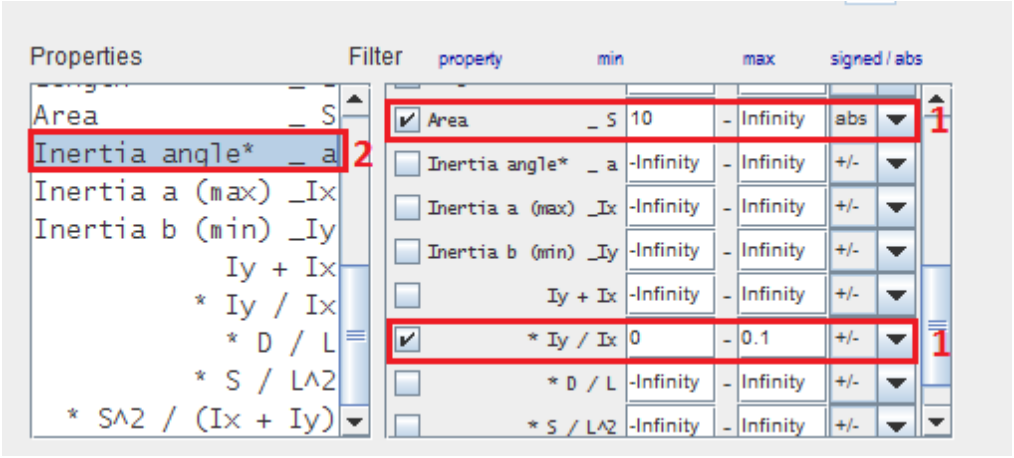

Рисунок 2.12 1 – свойства контуров, по которым будет осуществляться фильтрация, 2 – свойство контуров, для которого необходимо показать диаграмму распределения.

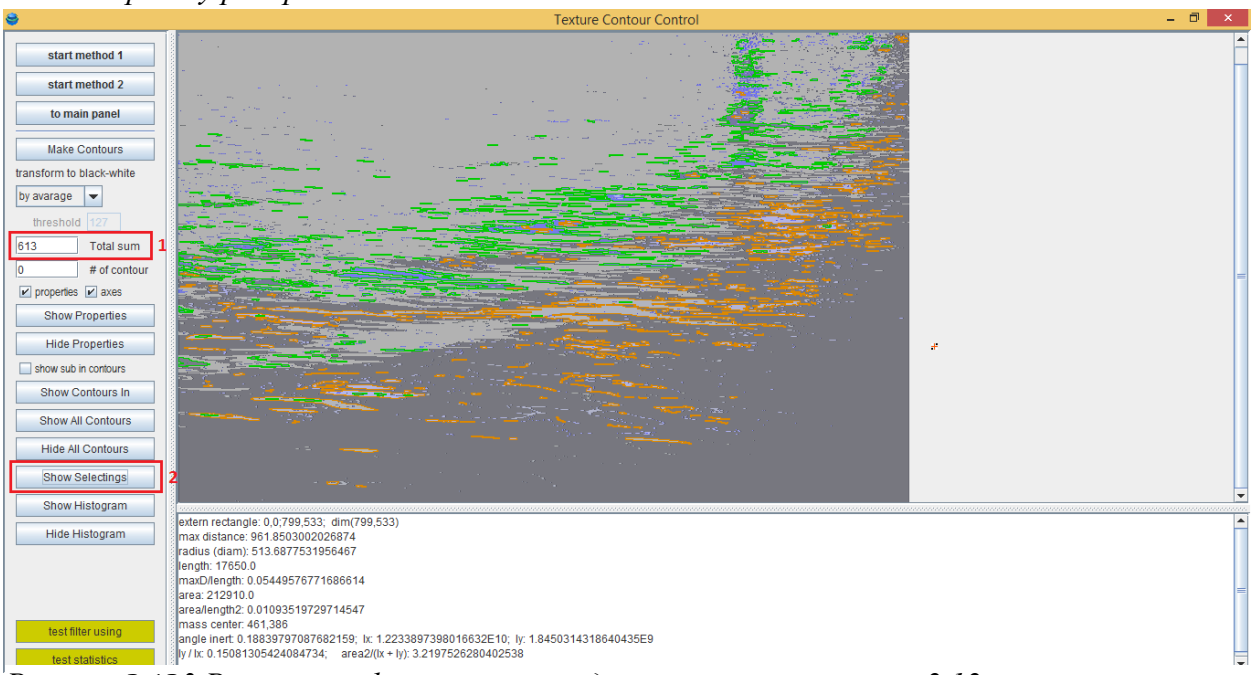

Рисунок 2.133 Результат фильтрации представленной на рисунке 2.12

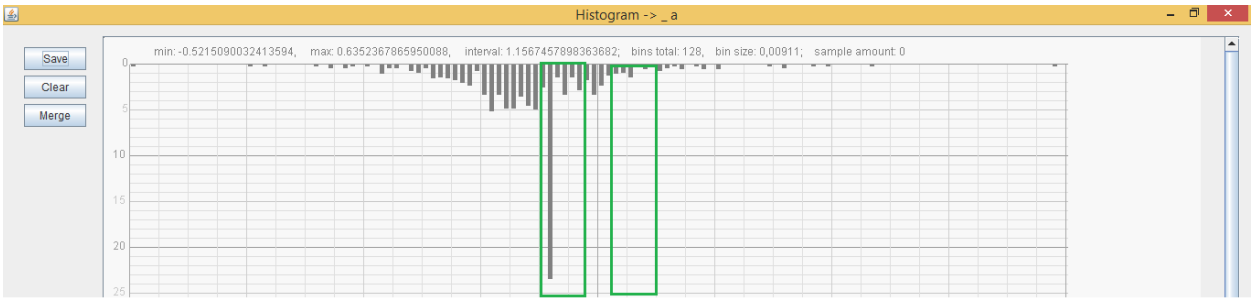

Рисунок 2.14 Гистограмма распределения контуров по углу инерции, где на вертикальной оси располагается количество контуров, на вертикальной угол инерции. Зелёная область – область контуров, которые считаются контурами водной поверхности (Исследование второе).

Выявив закономерность, по которой предположительно водная поверхность отличается от городской, возникла потребность в множественной

обработки фотоснимков. Для этого, была реализована программа на языке Java, в среде IntelIJ IDEA 15.0. Данное приложение работает с множеством изображений из указанной директории. Важной особенностью программы является то, что поток, в котором идёт работа с изображениями, можно приостановить. Главное окно программы представлено на рисунке 2.15. Нажимая на кнопку исходная папка, выберается папка, фото из которой необходимо обработать, также если необходимо, можно выбрать папку в которую будут сохранены изменения. Далее по нажатии на кнопку выполнить запускается обработка файлов, название копки меня есть с «Выполнить» на «Прекратить» (По нажатии на которую будет выполнена приостановка потока, до следующего нажатия на кнопку), в текстовом поле «Всего» отображается сколько всего файлов (изображений) в папке, в поле «Готово» обновляется количество обработаных фотоснимков.

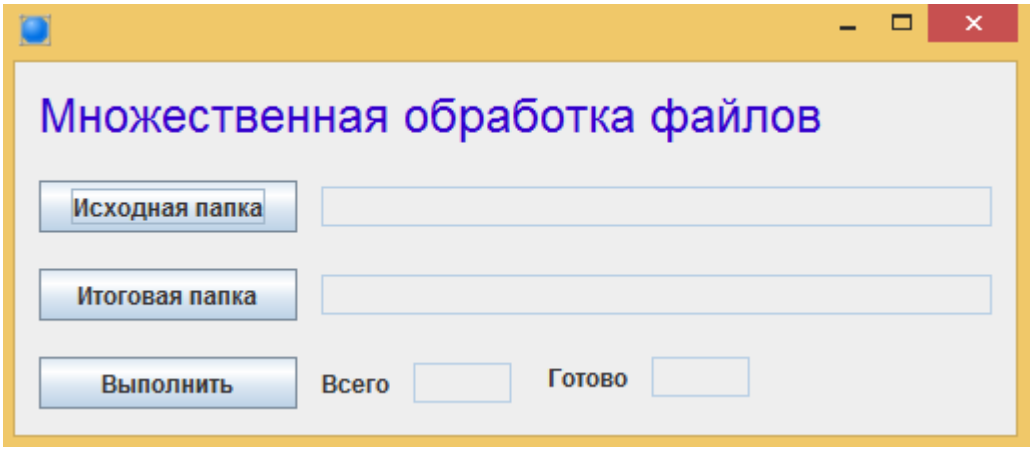

Рисунок 2.16 Главное окно программы для множественной обработки изображений

Глава 3. Классификация изображений

В данной работе рассматривается задача распознавания однородных изображений (т.е. таких фотоснимков, на большей части которых изображена одна тематика):

1. водной поверхности
2. городской местности
3. гор
4. равнинных полей
5. лесных массивов
6. пустыни.

Перед тем, как приступить к постановке задачи классификации, мы столкнулись с множеством трудностей, основные из них:

1) Количество контуров варьируется от изображения к изображению
2) У одного изображения множество контуров со схожими или даже одинаковыми характеристикам
3) Выбор подходящих параметров контуров
4) Поиск обучающей выборки

Первую и вторую задачи было принято решить разбиением контуров на определённые группы, это решит то, что каждый фотоснимок характеризуется разным набором параметров. Изображение, после кластеризации, будет представлять собой вектор размерностью, равной числу групп.

Третья задача является самой сложной, т.к. для различных изображений одни параметры подходят, для других нет, а хотелось бы, что бы один набор свойств подходил для всех тематик.

На решение четвёртой задачи было затрачено много времени, не нашлось сразу целой обучающей выборки на каком-либо ресурсе, так, как например для задачи распознавания автомобильных номеров или лиц людей, можно найти без особых проблем, т.к. данные вопросы исследуются более

активно. Приложив максимум усилий удалось найти выборку, содержащую 2990 изображений.

3.1 Кластерный анализ

Постановка задачи:

Рассмотрим множество объектов I (3.1), каждый из которых представляется набором атрибутов. Необходимо построить множество кластеров C (3.2) и отображение F : {F : I → C} [5].

$$I = \{i_1, i_2, \dots, i_{n-1}, i_n\} \quad (3.1)$$

$$C = \{c_1, c_2, \dots, c_{k-1}, c_k\} \quad (3.2)$$

$$c_k = \{i_j, i_p | i_j, i_p \in I \text{ и } d(i_j, i_p) < \sigma\} \quad (3.3)$$

c_k – кластер, который содержит в себе похожие объекты из I, степень похожести выбирается заданием $d(i_j, i_p) < \sigma$, где σ – величина определяющая степень близости объектов кластера, $d(i_j, i_p)$ – мера близости, расстояние между объектами, d – удовлетворяет всем свойствам меры, определенным в математическом анализе [5].

Таких образом, задача кластеризации – разбить множество данных на похожие группы, где близость объектов определяется заданием меры расстояния между векторами x_i и x_j размерности n, как x_{it} обозначается t-ая координата вектора x_i.

Основные меры расстояния:

a) Евклидово:

$$d_2(x_i, x_j) = \sqrt{\sum_{t=1}^{n}(x_{it} - x_{jt})^2} \quad (3.4)$$

b) Хемминга:

$$d_H(x_i, x_j) = \sum_{t=1}^{n}|x_{it} - x_{jt}|$$

c) Чебышева:

$$d_\infty(x_i, x_j) = \max_{1 \le t \le n}|x_{it} - x_{jt}|$$

Существует ещё множество используемых мер при кластеризации, с

ними можно познакомиться в работе [5]. В данной работе кластеризация применяется для контуров со свойствами, для расчёта расстояний между контурами использовалась Евклидова метрика.

Существует множество методов кластеризации: иерархические, неиерархические, дивизимные [5]. Наибольшую популярность имеют неиерархические методы, т.к. они находят оптимальное разбиение на кластеры, минимизируя некоторую выбранную целевую функцию разбиения. В текущей работе был выбран один из таких алгоритмов, *k-means*.

3.1.1 Кластеризация методом k-means

Архитектура модели описывается входным множеством, количеством кластеров, центрами кластеров, метрикой, целевой функцией.

- Обучающее множество размером векторов размерности n:

$$M = \{ m_1, \dots, m_d \}$$

- Метрика расстояния, вычисляется по формуле (3.4)
- Вектор центра кластеров C, каждая компонента которого задаётся формулой:

$$c^{(i)} = \frac{\sum_{j=1}^{d} u_{ij} m_j}{\sum_{j=1}^{d} u_{ij}}, i = 1 \dots c$$

- Матрица разбиения U =$\{u_{ij}\}$, где u_{ij} заданы формулой:

$$u_{ij}{}^{(l)} = \begin{cases} 1, \text{при } d\left(m_j, c_i{}^{(l)}\right) = \min\limits_{1 \le k \le c} d\left(m_j, c_k{}^{(l)}\right) \\ 0, \text{в остальных случаях} \end{cases} \qquad (3.5)$$

- Целевая функция:

$$J(M, U, C) = \sum_{i=1}^{c} \sum_{j=1}^{d} (u_{ij})\, d_A^2(m_j,\ c_i)$$

Алгоритм:

- **Шаг 0.** Инициализация начального разбиения, выбор точности δ, номера начальной итерации l=0.
- **Шаг 1.** Вычисление центров кластеров по формуле:

$$c_l{}^{(i)} = \frac{\sum_{j=1}^{d} u_{ij}{}^{(l-1)} m_j}{\sum_{j=1}^{d} u_{ij}{}^{(l-1)}}, i = 1 \dots c$$

- **Шаг 2.** Определение центров кластеров, для данного разбиения по формуле (3.5)
- **Шаг 3.** Проверка условия, если оно выполняется, то завершаем, иначе повтор шагов 1-3, пока условие не будет выполненным.

$$\left\| U^{(l)} - U^{(l-1)} \right\| < \delta$$

Таким образом на выходе из алгоритма будут получены центры кластеров. Каждый объект тестовой выборки относится к тому кластеру, расстояние до которого наименьшее.

3.2 Кластеризация контуров

Как было сказано ранее, для задачи классификации, необходимо, что бы все объекты обучающего и тестового множества были одной размерности (имели одинаковое число свойств). В случае с изображениями, число параметров варьируется от изображения к изображению. Поэтому было решено выделить группы контуров, и характеризовать фотоснимок, как вектор, каждая компонента которого содержит в себе количество контуров на текущем изображении, принадлежащих определённой группе. Выделение групп было решено делать с помощью кластеризации методом *k-means*.

Постановка задачи кластеризации контуров

Дано:

- W_1 – обучающее множество изображений
- W_2 – тестовое множество изображений
- Каждое изображение задаётся матрицей:

$$W^{(i)} = \begin{pmatrix} {w_{11}}^{(i)} & \cdots & {w_{1l}}^{(i)} \\ \vdots & \ddots & \vdots \\ {w_{v1}}^{(i)} & \cdots & {w_{vl}}^{(i)} \end{pmatrix}$$

Строка матрицы – контур, столбец – свойства контура.

Необходимо:

- Определить число *n* групп (кластеров) на которые контуров обучающего множества
- Выполнить кластеризацию контуров и найти центры кластеров
- Распределить контуры изображений тестового множества по получившимся кластерам
- Представить каждое изображение в новом виде, где $k_j^{(i)}$ – число контуров на i-ом изображении попавших в j-ый кластер, где n – число кластеров:

$$W^{(i)} = (k_1^{(i)} \quad k_2^{(i)} \quad \ldots \quad k_{n-1}^{(i)} \quad k_n^{(i)})$$

3.2.1 Программная реализация

Для определения числа кластеров использовался критерий Калинского – Харабаза [11]. Для реализация данного метода использовался пакет программного продукта MATLAB Statistics and Machine Learning Toolbox, пример работы функции, реализующей данный метод Калинского – Харабаза представлен на рисунке 3.1, в первой строке которого описан вызов метода кластеризации *k-means* в MATLAB с параметрами, X – данные для кластеризации, K – число кластеров. Так же указывались такие параметры, как Distance – используемая метрика (была выбрана Евклидова), Options – настройки метода, в данном случае было увеличено максимальное число

итераций кластеризации до 1000 (В MATLAB по умолчанию 100). Во второй строке вызов функции для определения оптимального числа кластеров, для данных ClasterCDFFMO. В 3-ем аргументе указан метод, которым осуществляется оптимальный поиск (в MATLAB представлены и другие метода), в текущем случае, это Калинского – Харабаза. Последний параметр KList – это интервал, с вариантами числа кластеров, которые необходимо перебрать (в данном случае от 10 до 40).

```
1    Fun = @(X,K)kmeans(X,K,'Distance','sqeuclidean','Options',statset('MaxIter',1000));
2    evaCDFFMO = evalclusters(ClasterCDFFMO,Fun,'CalinskiHarabasz','KList',[10:40]);
```

Рисунок 3.1 Пример вызова функции evalcluster, реализующей поиск оптимального числа кластеров, в MATLAB. В первой строке определяется метод для кластеризации, в данном случае k-means. Во второй вызов функции evalcluster с параметрами: ClasterCDFFMO – кластеризуемые данные, Fun – метод которым осуществляется кластеризация (с параметрами: Евклидова мера расстояния и максимальное число итераций 1000), CalinskiHarabasz – метод поиска оптимального числа кластеров, Klist – массив значений в пределах которого необходимо найти оптимальное число кластеров.

После того, как найдено оптимальное число кластеров, осуществляется кластеризация, пример кластеризации на рисунке 3.2. Где X – первый выходной параметр метода, содержащий центры кластеров, получившиеся из *k-means*, Y – вектор с номерами кластеров для каждого объекта данных. Получив центры кластеров X, необходимо распределить все контуры обучающей и тестовой выборки по группам в соответствии с тем, к какому из кластеров контур ближе. Данная работа была выполнена в программе, описанной в параграфе 2.3 и изображённой на рисунке 2.15. Реализация представлена на рисунках 3.3-3.6. На первом рисунке представлен метод, принимает папку с текстовыми файлами, в которых в свою очередь описаны всё изображения в виде контуров, возвращает список в котором хранятся изображения, представленные в новом виде. В 63 строке этого метода вызывается функция clastering (Рисунок 3.4). Метод принимает контур изображения, возвращает номер кластера, к которому относится данных контур. На рисунке 3.5 метод, для вывода изображений в новом формате в файл (Это необходимо для дальнейшей работы, т.к. классификацию

изображений будем осуществлять в MATLAB). На рисунке 3.6 можно увидеть пример вызова функций.

```
14
15      [X Y] = kmeans(ClasterCDFFMO,15,'Distance','sqeuclidean','Options',statset('MaxIter',1000));
```

Рисунок 3.2 Кластеризация методом k-means в среде MATLAB, с оптимальным числом кластеров 15 (Найденным ранее) и данными ClasterCDFFMO. X и Y – выходные параметры (Центры кластеров и номера кластеров для каждого объекта данных соответственно)

```
35      public static List<List<double[]>> fileReader(String filesPath, String count) {
36          List<List<double[]>> list2 = new ArrayList<List<double[]>>();
37          List<double[]> list = new ArrayList<double[]>();
38          List<double[]> listOuters = new ArrayList<double[]>();
39          String s;
40          String[] s1;
41
42          File folder = new File(filesPath);
43          File[] folderEntries = folder.listFiles();
44          for (File entry : folderEntries) {
45              if (entry.isDirectory() && (entry.getAbsolutePath().toString().contains(count))) {
46                  File folder2 = new File(entry.getAbsolutePath());
47                  File[] folderEntries2 = folder2.listFiles();
48                  for (File file : folderEntries2) {
49                      try (FileReader reader = new FileReader(file.getAbsolutePath())) {
50                          if (file.getAbsoluteFile().toString().contains(".txt")) {
51                              int t = 0;
52                              BufferedReader in = new BufferedReader(reader);
53                              //число кластеров
54                              double[] pic = new double[12];
55                              double[] d1 = new double[1];
56                              while ((s = in.readLine()) != null) {
57                                  s1 = s.split(" ");
58                                  //число свойств
59                                  double[] d = new double[5];
60                                  for (int i = 0; i < 5; i++) {
61                                      d[i] = Double.valueOf(s1[i]);
62                                  }
63                                  pic[clastering(d) - 1]++;
64                                  //последний элемент вектора контура
65                                  d1[0] = Double.valueOf(s1[15]);
66                              }
67                              list.add(pic);
68                              listOuters.add(d1);
69                          }
70                      } catch (IOException ex) {
71
72                          System.out.println(ex.getMessage());
73                      }
74                  }
75                  continue;
76              }
77          }
78          list2.add(list);
79          list2.add(listOuters);
80          return list2;
81      }
```

Рисунок 3.3 Метод принимающий путь до папки с текстовыми файлами (в каждом из которых изображение записано, как набор вектором – контуров) обучающей и тестовой выборок. На выход из методе из метода получаем List – с изображениями, представленными в новом виде.

```
83      public static int clastering(double[] d) {
84          double[][] A ={
85                  {-0.0586304350565949,   0.321941139954265,  0.284603832777451,  0.420569678141513,  4.14011749257211},
86                  {-0.0527688980691550,   0.185380167567149,  0.293896637171978,  0.337185824815500,  3.11962662690963},
87                  {-0.0470708808844448,   0.0228749107771907, 0.365066307927531,  0.183305819454455,  1.09768182219544},
88                  {0.0168563598300082,    0.0833199491931675, 0.320990402714432,  0.275409364027519,  2.15515359165524},
89                  {1.13088674840120,  0.154540806037146,  0.299567513488913,  0.339132834432267,  2.93167999317672},
90                  {-1.14066801922606, 0.439123617970088,  0.288439713523714,  0.530612251421945,  4.87139528037591},
91                  {1.02899525097991,  0.473330631339565,  0.289644616245799,  0.546076938185787,  5.03794705375748},
92                  {-1.23981215041390, 0.0870136590448595, 0.317020713623081,  0.282983852342008,  2.18895495416401},
93                  {1.25333497876591,  0.0592222136291746, 0.319057854214627,  0.223918512252921,  1.72483614064101},
94                  {-0.117630023833510,    0.503476309119155,  0.296208801692096,  0.610485051406176,  5.43785383440113},
95                  {1.11558788492293,  0.278997928297647,  0.289621212517569,  0.418675633727105,  3.89633614878703},
96                  {-1.21397718167655, 0.206803455429122,  0.306157986789350,  0.431333467445262,  3.55585497041219}};
97          double min = 1E+7;
98          int t = 0;
99          //число кластеров
100         for (int j = 0; j < 12; j++) {
101             double sum = 0;
102             //число свойств
103             for (int i = 0; i < 5; i++) {
104                 sum += (A[j][i] - d[i]) * (A[j][i] - d[i]);
105             }
106             sum = Math.pow(sum, 0.5);
107             if (sum < min) {
108                 min = sum;
109                 t = j + 1;
110             }
111         }
112         return t;
113     }
```

Рисунок 3.4 Метод осуществляющий распределение контуров по кластерам, принимает контур (массив), возвращает номер ближайшего кластера для данного контура.

```
115     public static void fileWriter(List<double[]> list, List<double[]> list2, String s) {
116         try (FileWriter writer = new FileWriter(s, true)) {
117             int k = 0;
118             for (double[] a : list) {
119                 for (int i = 0; i < a.length; i++) {
120                     writer.write(a[i] + " ");
121                 }
122                 writer.write((Double.valueOf(list2.get(k)[0])).toString());
123                 writer.append("\r\n");
124                 k++;
125             }
126         } catch (IOException ex) {
127
128             System.out.println(ex.getMessage());
129         }
130     }
131 }
```

Рисунок 3.5 Метод для вывода данных в файл.

```
6       public static void main(String[] args) {
7
8           List<List<double[]>> LList = fileReader("D:\\Data\\DataSet", "Train");
9           List<double[]> list = new ArrayList<double[]>();
10          list = LList.get(0);
11          List<double[]> list2 = new ArrayList<double[]>();
12          list2 = LList.get(1);
13
14          List<List<double[]>> LListTest = fileReader("D:\\Data\\DataSet", "Test");
15          List<double[]> listTest = new ArrayList<double[]>();
16          listTest = LListTest.get(0);
17          List<double[]> listTest2 = new ArrayList<double[]>();
18          listTest2 = LListTest.get(1);
19
20          fileWriter(list, list2, "D:\\Data\\NuronDataSet\\Train12.txt");
21          fileWriter(listTest, listTest2, "D:\\Data\\NuronDataSet\\Test12.txt");
22      }
23
```

Рисунок 3.6 Метод –точка входа в программу. На изображении представлен пример вызова метода, переводящего изображение в новое представление (в виде вектора).

Для кластеризации (определения групп и центров кластеров) было решено взять по 300 изображений каждой тематики из перечисленных в начале 3 главы, на них было найдено 1055646 контуров. Были рассмотрены несколько различных групп свойств контуров (более подробно свойства описаны в параграфе 1.2):

1) Инвариантные к равномерному растяжению и повороту
   a) R_I – отношение моментов инерции
   b) R_D – отношение максимального радиуса к длине контура.
   c) R_L – отношение площади фигуры, охватываемой контуром к квадрату длины контура
   d) R_S – Отношение квадрата площади к полярному моменту
   e) β (данное свойство не является инвариантным, но т.к. с его помощью хорошо различалась водная поверхность от города, то его решено было добавить в данную группу).
2) Не инвариантные
   a) D_x – горизонтальный диаметр контура
   b) D_y – вертикальный диаметр
   c) D – максимальны диаметр контура (удвоенный радиус)
   d) S – Площадь контура
   e) L – Периметр контура
   f) β – угол инерции контура

Что бы понять, какие параметры для кластеризации использовать, были перебраны многие варианты сочетаний свойств контуров. Оказалось, что группа неинвариантных параметров давала очень плохую кластеризацию, оказалось, что большая часть контуров попадает в один кластер. Таким образом была отброшена одна группа свойств. Вторая группа давала почти равномерное распределение контуров по кластерам, поэтому заострили внимание на ней. Для того, чтобы определить по каким из свойств стоит

выделять группы контуров, были перебраны все варианты сочетаний свойств, для каждого находилось оптимальное число кластеров в интервала от 10 до 40, методом Калинского – Харабаза. Отрывок кода реализации данной процедуры представлен на рисунке 3.7. Таким образом мы пытались найти кластеризацию с самым равномерным разделением данных по кластерам, дающую наибольшее число кластеров, при наибольшем числе параметров, т.к. большее число параметров даёт больше информации о контуре, а наибольшее число кластеров даёт наибольшее число групп. Лучшие результаты получились для сочетаний свойств:

1) β, R_S – 30 кластеров
2) R_I, R_S – 35 кластеров
3) R_I, R_D, R_L, R_S, β – 14 кластеров (самый информативный набор)

После того, как определились лучшие группы свойств и оптимальное число кластеров, проделаем кластеризацию (Рисунок 3.8). Полученные центра кластеров (Переменные YCDMFFMO25, YCDMFFMO15, YCDMFFMOAll) перенесём в программу для работы с множеством изображений (т.е. определим массив double[][] A в методе clastering – рисунок 3.4). После этого разложим контуры изображений тестового и обучающего множеств по кластерам. Запустив программу из main (Рисунок 3.6). Проделав так, для каждой из трёх кластеризаций, получим 3 файла обучающими данными, и 3 файла с тестовыми.

```
10 -    Fun = @(X,K)kmeans(X,K,'Distance','sqeuclidean','Options',statset('MaxIter',1000));
11 -    evaCDFFMO1 = evalclusters(ClasterCDFFMO(:,1),Fun,'CalinskiHarabasz','KList',[10:40]);
12 -    evaCDFFMO2 = evalclusters(ClasterCDFFMO(:,2),Fun,'CalinskiHarabasz','KList',[10:40]);
13 -    evaCDFFMO3 = evalclusters(ClasterCDFFMO(:,3),Fun,'CalinskiHarabasz','KList',[10:40]);
14 -    evaCDFFMO4 = evalclusters(ClasterCDFFMO(:,4),Fun,'CalinskiHarabasz','KList',[10:40]);
15 -    evaCDFFMO5 = evalclusters(ClasterCDFFMO(:,5),Fun,'CalinskiHarabasz','KList',[10:40]);
16
17
18 -    evaCDFFMO12 = evalclusters(ClasterCDFFMO(:,1:2),Fun,'CalinskiHarabasz','KList',[10:40]);
19 -    evaCDFFMO13 = evalclusters(cat(2,ClasterCDFFMO(:,1),ClasterCDFFMO(:,3)),Fun,'CalinskiHarabasz','KList',[10:40]);
20 -    evaCDFFMO14 = evalclusters(cat(2,ClasterCDFFMO(:,1),ClasterCDFFMO(:,4)),Fun,'CalinskiHarabasz','KList',[10:40]);
21 -    evaCDFFMO15 = evalclusters(cat(2,ClasterCDFFMO(:,1),ClasterCDFFMO(:,5)),Fun,'CalinskiHarabasz','KList',[10:40]);
```

Рисунок 3.7 Пример поиска оптимального числа кластеров на данных ClasterCDMFFO (контура с инвариантными свойствами). 10-15 строки – поиск по одному свойству, 18-21 поиск по всем вариантам сочетаний 2 свойств, одно из которых 1-ое (угол инерции)

```
Data25=cat(2,ClasterCDFFMO(:,2),ClasterCDFFMO(:,5));
Data15=cat(2,ClasterCDFFMO(:,1),ClasterCDFFMO(:,5));
[XCDFFMO25 YCDFFMO25] = kmeans(Data25,35,'Distance','sqeuclidean','Options',statset('MaxIter',1000));
[XCDFFMO15 YCDFFMO15] = kmeans(Data15,30,'Distance','sqeuclidean','Options',statset('MaxIter',1000));
[XCDFFMOAll YCDFFMOAll] = kmeans(ClasterCDFFMO,14,'Distance','sqeuclidean','Options',statset('MaxIter',1000));
```

Рисунок 3.8 Кластеризация данных по лучшим сочетаниям свойств. Data25 – множество контуров со свойствами R_I и R_S, Data15 – β и R_S, ClasterCDFFMO – данные по всем свойствам из группы инвариантных свойств.

После того, как была проделана кластеризация контуров изображений, каждое изображение стало представляться, как вектор (ранее оно задавалось как матрица). Теперь можно без сложностей, которые были ранее, составить задачу классификации.

3.3 Нейронные сети

Постановка задачи:

Дано:

- W_1 – обучающее множество изображений, Y_1 – значения классов к которым принадлежит каждое изображение данного множества

$$W_1 = \begin{pmatrix} k_{11} & \cdots & k_{1n} \\ \vdots & \ddots & \vdots \\ k_{m1} & \cdots & k_{mn} \end{pmatrix}, Y_1 = \begin{pmatrix} y_{11} \\ \vdots \\ y_{1m} \end{pmatrix}$$

- W_2 – тестовое множество изображений

$$W_2 = \begin{pmatrix} k_{11} & \cdots & k_{1n} \\ \vdots & \ddots & \vdots \\ k_{k1} & \cdots & k_{kn} \end{pmatrix}$$

i-ая строка матрицы – изображение, j-ый столбец – характеристика контура (число контуров попавших в кластер с номером j).

Необходимо:

- Обучить классификатор на множестве изображений W_1.
- Добиться максимальной точности, на изображениях тестового множества W_2

Для решения поставленной задачи использовалась классическая нейронная сеть типа персептрон, с одним скрытым слоем.

3.3.1 Персептрон

Понятие персептрона.

Впервые данное понятие было введено Френком Розенблеттом в 1957 году, как класс моделей мозга [12]. Схематически персептрон представлен на рисунке 3.9. S – множество чувствительных элементов, A – множество ассоциирующих нейронов, R – реагирующий элемент. Нейрон имеет много входов и один выход. Входы бывают тормозящие и возбуждающе. S-элементы возбуждаются, если при воздействии раздражителя входной сигнал будет больше порогового значения. Такие элементы связаны случайным образом с A-нейронами. Если число возбуждающих сигналов больше числа тормозящих, то A-нейрон возбуждается и передает сигнал на R-элемент. Сигналы, поступающие на реагирующий элемент, суммируются с некоторыми весами. R-элемент выбирает некоторое действие, если [12]:

$$R(x) = \sum_{i=0}^{n} x_i\, w_i = (w, x) > 0$$

Где $w=(w_0, w_{1,\dots,} w_{n-1,} w_n)$, $x=(x_0, x_{1,\dots,} x_{n-1,} x_n)$, x_i – сигнал, поступающий на R-элемент i-ого нейрона, x_0 – сигнал смещения нейрона, он задаётся при инициализации нейронной сети перед обучением, и далее не изменяется. Суть персептрона: 1) на вход подаются обучающие примеры (входной слой), 2) затем каждый элемент входного слоя подаётся в каждый нейрон скрытого слоя (рисунок 3.10), проходя через функцию активации f_i (рисунок 3.10), 4) далее от скрытого к скрытому, до тех пор, пока слои не закончатся, 5) заключительный этап – сигналы последнего скрытого слоя поступают на каждый нейрон выходного слоя с определёнными весами.

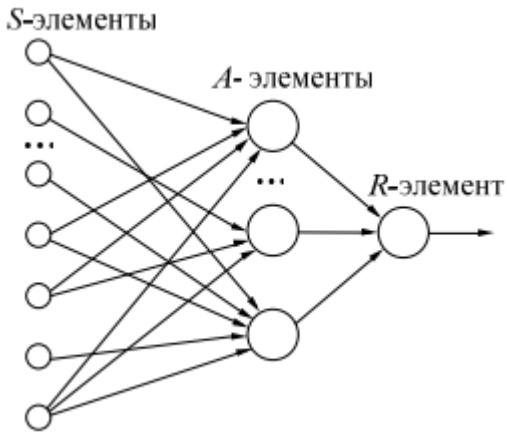

Рисунок 3.9 Персептрон

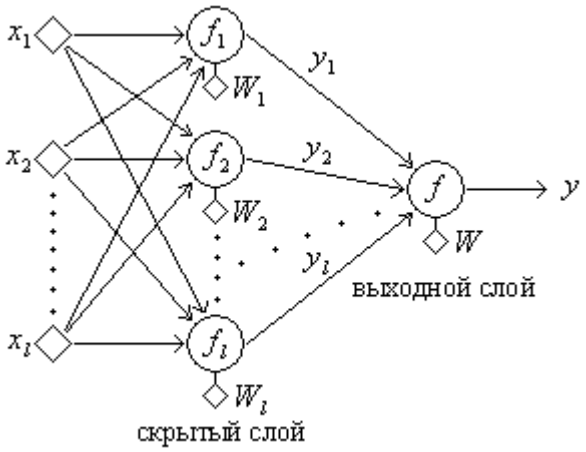

Рисунок 3.10 Персептрон со скрытым слоем

Алгоритм обучение персептрона

Пусть имеется обучающая выборка X = {x_1,…, x_n} и m классов, по которым необходимо классифицировать объекты обучающего множества.

1) Инициализация всех весовых коэффициентов, задание функции активации, задание функции ошибки, задание
2) Начиная с последнего слоя персептрона (в случае обратного распространения ошибки), вычисляются новые весовые коэффициенты, данную процедуру можно сделать различными способами:
    a) Случайным изменением весов
    b) Квазианалитическим способом
    c) Минимизацией функцию ошибки
3) Проверка условия останова, обычно накладывается ограничение на функцию ошибки.

В данной работе рассматривается персептрон с одним скрытым слоем, с помощью сети с такой архитектурой можно решит любую задачу обучения, для любой наперёд заданной точности [12]. Начальный выбор весов определяется случайным образом. В качестве функции активации была выбрана сигмоидальная функция (3.7). Функция ошибки – среднеквадратическая, рассчитывается по формуле (3.8), где Y_i – наперёд

известное значение, $\overline{Y}_i$ – значение на выходе сети. Обучение происходи методом обратного распространения ошибка, т.е. изменение весовых коэффициентов начинается с выходного слоя, затем проходит по всем скрытым. Минимизация функции ошибки происходит методом градиентного спуска, с использованием одномерной минимизации золотым сечением.

$$f(x) = \frac{1}{(1-e^{-ax})} \tag{3.7}$$

$$E = \frac{1}{n}\sum_{i=1}^{n}(\overline{Y}_i - Y_i)^2 \tag{3.8}$$

3.3.2 Классификация изображений нейронной сетью

Как было сказано в параграфе 3.2.1 отобралось 3 варианта лучших наборов свойств:

1) β, R_S – 30 кластеров
2) R_I, R_S – 35 кластеров
3) R_I, R_D, R_L, R_S, β – 14 кластеров

После проделанной кластеризации получили 3 набора данных – это данные обучающего и тестового набора изображений, каждый из которых прокластеризован по 3 наборам свойств. Имея набор данных 2990 изображений, каждое из множеств. Каждый набор разбили на 3 множества, обучающее, тестовое и валидационное 2092, 784 и 150 фотоснимков соответственно. Для каждого набора была обучена нейронная сеть в среде MATLAB, используя Neural Network Toolbox. Данный пакет очень удобен в использовании, написав команду nnstart в консоли MATLAB, будет вызван пользовательский интерфейс, в которым осуществлялась основная работа с нейронной сетью. Рассмотрим пример полученной нейронной сети на данных полученных по свойствам контуров R_I, R_D, R_L, R_S, β (рисунок 3.11). Сети с лучшей точностью представлены на рисунке 3.12, их описание в таблице 3.1. Наилучшая точность была получена нейронной сетью на данных полученных кластеризацией по параметрам β, R_S. Так же в конце хотелось проверить какая точность будет, при использовании других классификаторов. В таблице 3.2

можно наблюдать результаты, полученные методами ближайшего соседа и деревом решений [12], в сравнении с результатами нейронной сети. Лучше всего показали себя нейронная сеть и метод ближайшего соседа на данных полученных кластеризацией по параметрам β, R_S.

Подробно о сети изображённой на рисунке 3.11:

1) Данные
    a) Обучающее множество 2092 изображений
    b) Тестовое множество 748 изображений
    c) Валидацонное множество 150 изображений
2) Архитектура сети
    a) Входной слой – 14 нейронов. Число входов
    b) Скрытый слой – 10 нейронов (С сигмоидальными нейронами)
    c) Выходной слой 6 нейронов (С линейными нейронами)
3) Используемые методы
    a) Выбор начальный значений весовых коэффициентов нейронов – случайный
    b) Обучение методом обратного распространения ошибки
    c) Функция ошибки – среднеквадратичная
    d) Минимизация функции ошибки методом градиентного спуска
4) Итоги обучения
    a) Метод обучился за 76 эпох
    b) Время обучения 14 секунд
    c) Значение функции ошибки: 0.0790
    d) Значение градиента функции ошибки: 0.00138

5) Точность сети

a) Обучающее множество 64%

b) Тестовое множество 62%

c) Валидационное множество 58%

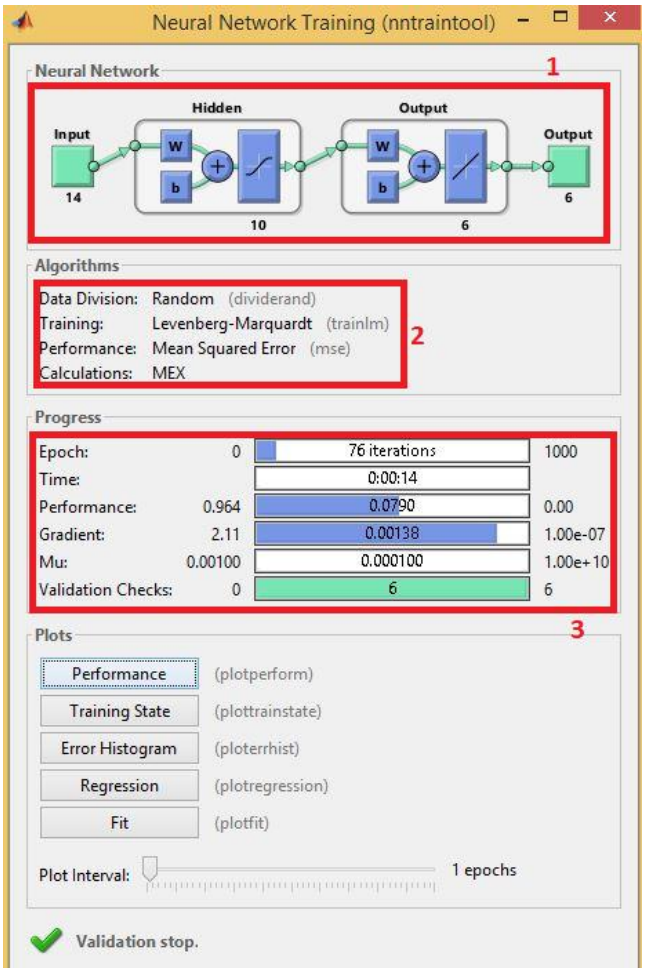

Рисунок 3.11 1 – архитектура сети, 2 - используемые методы, 3 – итоги обучения сети

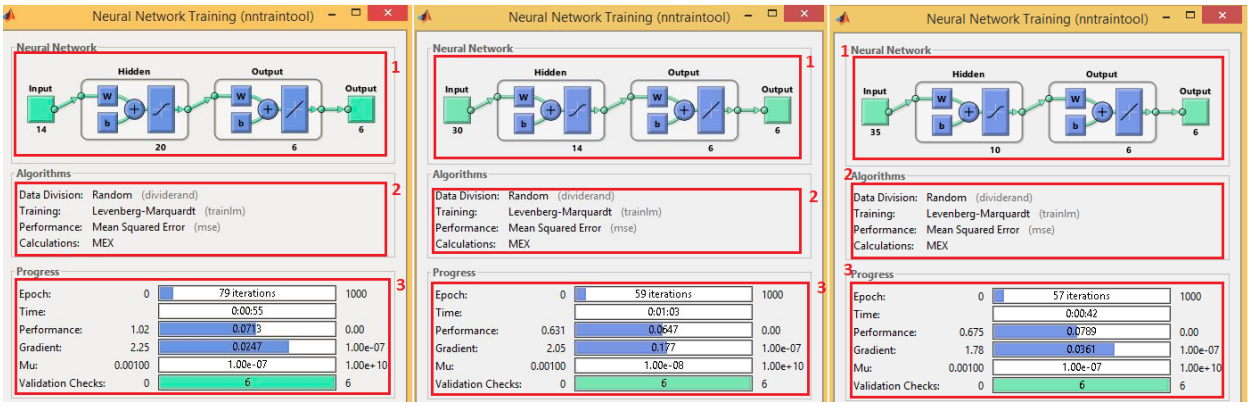

Рисунок 3.12 Три сети с лучшими результатами. Слева – сеть, полученная кластеризацией по параметрам R_I, R_D, R_L, R_S, β. По центру – R_S, β. Справа – R_S, R_I.

Таблица 3.1 Результаты работы нейронных сетей, полученных на 3 наборах данных: R_I, R_D, R_L, R_S, β; β, R_S; R_I, R_S соответственно для 2-4 столбцов таблицы.

Данные		**14 кластеров**	**30 кластеров**	**35 кластеров**
Число нейронов в скр. слое		20	14	10
Точность (%)	Обучающее мн-во.	64	72	65
	Тестовое мн-во.	**62**	**67**	**61**
	Валидац. мн-во.	58	65	62
Число эпох обучения		79	59	57
Время обучения (сек.)		55	103	42
Значение функции ошибки		0.0713	0.0647	0.0789
Значение градиента		0.0247	0.177	0.0361

Таблица 3.2 Точность распознавания на 3 наборах данных различными методами классификации.

Данные / **Методы классификации**	**14 кластеров (Точность %)**	**30 кластеров (Точность%)**	**35 кластеров (Точность%)**
Персептрон	62	**67**	61
Дерево решений	59	60	61
Метод ближайших соседей	63	**67**	46

3.4 Существующие решения

В таблице 3.3 представлены результаты сравнения с существующими решениями задачи, рассмотренной в данной работе. Для сравнения были взяты алгоритмы, построенные на свёрточных нейронных сетях. Первый алгоритм, предложенный Massachusetts Institute of Technology (Массачусетском

Университетом Технологий) – Places2. В этом проекте рассматривается задача определения местности на фотоснимке. Второй, это продукт Microsoft CaptionBot. Если первый проект предназначен для определения местности, то второй, решает множество других задач, таких как поиск людей на фото, определения его эмоций, поиск знаменитостей и др. Оба проекта на данный момент представлены в demo-версиях. Для сравнения было взято по 20 изображений каждой тематики из рассматриваемых в данном исследовании (всего тематик 6). В таблице 3.3 можно видеть, что точность распознавания обоих алгоритмов, на цветных изображениях около 80%, а на чёрно-белых около 50%. То, что точность хуже на чёрно-белых изображениях следует из того, что представленные алгоритмы работают с цветом изображения. В подходе к распознаванию, предложенном в данной работе, нет привязки к цвету, как было сказано ранее. Если судить о точности распознавания цветных изображений, то подход уступает на 25% предложенным алгоритмам, если же о точности распознавания чёрно-белых фотоснимков, то превышает на 15%.

Таблица 3.3 Сравнение точности методов, основанных на свёрточной нейронной сети

Данные	Точность (%) распознавания. (цветное изображение/черно-белое)	
	Places2	**CaptionBot**
город	95/80	80/60
пустыня	85/15	50/25
поле	90/70	75/30
вода	70/25	100/50
горы	55/40	100/70
лес	95/85	90/70
Средняя точность	**81/52**	**82/51**

Заключение

Распознавание изображений, с помощью анализа свойств контуров, показало хорошие результаты. Максимальная точность, которую получилось достичь 67%. В сравнении с существующими решениями данной задачи, подход показал себя неплохо, если учесть то, что метод работает только с изображениями в бинарном виде. Если же сравнивать с методами, которые учитывают цвет изображения, то алгоритм предложенный в текущей работе проигрывает существующим.

Плюсы подхода:

1) Высокая скорость обучения – это обусловлено тем, что каждое изображение характеризуется малым числом параметров.
2) Нет привязки к цвету фотоснимка или к аппарату его снимающего, т.к. алгоритм проводит бинаризацию.
3) Низкая требовательность к памяти хранения данных, из-за того, что метод работает только с изображениями в бинарном виде, тогда каждый пиксель изображения может быть лишь двух вариантов цветов.

Минусы подхода:

1) Полученная точность на 14% ниже точности, предлагаемой существующими решениями.
2) Сложность выбора свойств контуров для распознавания – вытекает из того, что фотографии могут быть полученными различными способами, с различных ракурсов. И для того, чтобы подобрать нужное свойство, необходимо досконально изучить как можно больше вариантов съёмок.

Список литературы

1. Введение в контурный анализ; приложения к обработке изображений и сигналов / Под редакцией Я.А. Фурмана. М.: Физматлит, 2003. 592 с.
2. Гонсалес Р., Вудс Р. Цифровая обработка изображений. М.: Техносфера, 2005. 1072 с.
3. Гернет М. М., Ратобыльский В. Ф. Определение моментов инерции. М.: Машиностроение, 1969. 274 с.
4. В.Т. Фисенко, Т.Ю. Фисенко, Компьютерная обработка и распознавание изображений: учеб. пособие. СПб: СПбГУ ИТМО, 2008. 192 с.
5. Барсегян А.А. Анализ данных и процессов: учебное пособие СПБ.: БХВ-Петербург, 2009. 512 с.
6. Немков Р.М. Метод синтеза параметров математической модели сверточной нейронной сети с расширенным обучающим множеством // Современные проблемы науки и образования, 2015. № 1-2.
7. Федоренко Ю.С. Технология распознавания образов с использованием свёрточной нейронной сети // Молодежный научно-технический вестник 2013. № 12.
8. Друки А.А., Милешин М.А. Алгоритмы распознавания рукописных подписей на основе нейронных сетей // Фундаментальные исследования, 2014. № 11-9. С. 1906-1910.
9. Павлов Н.Н., Степанов А.П. Распознавание образов на основе сверточной сети для операционной системы андроид // Аммосов-2014 Сборник материалов всероссийской научно-практической конференции, проводимой в рамках Форума научной молодежи федеральных. университетов Северо-Восточный федеральный университет им. М.К. Аммосова / Под ред. Н.В. Малышевой. Киров: МЦНИП, 2014. С. 1089-1093.
10. Соладтова О.П. Применение сверточной нейронной сети для

распознавания рукописных цифр // Компьютерная оптика, 2010. №2.
11. Calinski R. B., Harabasz J. A dendrite method for cluster analysis // Communications in Statistics. 1974. Vol. 3, No 1. P. 1-27.
12. Лепский А.Е., Броневич А.Г. Математические методы распознавания образов: Курс лекций. Таганрог: ТТИ ЮФУ, 2009. 155 с.